OpenSFS Lustre Development

SCHEDULE OF ARTICLES
FOR
CONTRACT NO. SFS-DEV-001

ARTICLE 1 – SCOPE OF WORK

A. The Contractor shall perform the OpenSFS Lustre Development further described in the Statement of Work incorporated in this Contract.

B. The Contractor shall address the projects described herein as well as the milestone-tasks, which are the units of work, and the milestones and deliverables described in the *Milestones & Deliverables* section of this document.

C. The Contractor shall furnish all personnel, supervision, materials, supplies, equipment, tools, facilities, transportation, testing, and other incidental items and services necessary for performance of the work. The Contractor shall deliver the materials, products, supplies, reports and residuals, as specified.

D. Acceptance of the work under this Contract shall be based on the Contractor's performance and completion of the work in consonance with high professional standards and compliance with the delivery and reporting requirements specified herein.

Project 1. Single Server Metadata Performance Improvements

Technical Description and Approach

Two major strategies for scaling Lustre metadata throughput are horizontal and vertical scale. Vertical scaling aims to increase the performance of a single-node server. Horizontal scaling aims to allow multiple server nodes to collaborate to improve the performance of a single file system. We propose projects to improve metadata performance addressing both of these areas. This first project is addressed at improving the vertical scalability of a server by improving software efficiency on existing nodes and allowing Lustre to take advantage of increasingly more powerful nodes as they become available.

Contractor engineers have put substantial effort into improving vertical scaling of Lustre, demonstrating significant gains through this approach, and developing considerable expertise in this area. Previous efforts improved Symmetric Multi-Processing (SMP) scaling for Lustre by reducing lock contention in the networking and remote procedure call (RPC) layers. However, these improvements cannot be fully realized while SMP scaling issues in the underlying file system implementation remain.

OpenSFS Lustre Development

This project consists of two subprojects for improving SMP scaling in the underlying file system as described below.

Subproject 1.1: SMP Node Affinity

This subproject splits the computing cores available on the Metadata Server (MDS) into a configurable number of compute partitions, and binds the Lustre RPC service threads to run within a specific compute partition. This allows the RPC threads to run more efficiently by keeping data structures in cache memory close to the CPU cores on which they are running, and avoids needless contention on the inter-CPU memory subsystem. SMP Node Affinity also allows individual RPC requests to stay local to a specific compute partition, improving overall efficiency throughout the protocol stack as the number of cores increases.

Subproject 1.2: Parallel Directory Operations

This subproject allows multiple RPC service threads to operate on a single directory without contending on a single lock protecting the underlying directory in the ldiskfs file system. Single directory performance is one of the most critical use cases for HPC workloads as many applications create a separate output file for each task in a job, requiring hundreds of thousands of files to be created in a single directory within a short window of time. Currently, both filename lookup and file system-modifying operations such as create and unlink are protected with a single lock for the whole directory.

This subproject will implement a parallel locking mechanism for single ldiskfs directories, allowing multiple threads to do lookup, create, and unlink operations in parallel. In order to avoid performance bottlenecks for very large directories, as the directory size increases, the number of concurrent locks possible on a single directory will also increase.

Technical Debt

The SMP Node Affinity functionality is also useful for removing scalability bottlenecks on Object Storage Server (OSS) nodes to improve input/output (I/O) efficiency, in addition to the benefits for MDS nodes. As well, the Parallel Directory Operations can have a substantial performance benefit for Object Storage Target (OST) file systems, where there are many millions of objects stored on each OST or in cases of high contention on a small part of the object namespace, due to the fact that the object namespace is relatively flat.

To avoid an ongoing maintenance burden with the integration of Parallel Directory Operations into the ldiskfs code, as was experienced with previous work in this area, the new Parallel Directory Operations code will largely be abstracted from the underlying ldiskfs implementation. While some close integration with the ldiskfs code is required, the goal of this project is on long-term maintainability and seeks to minimize this boundary.

Project 2. Distributed Namespace: Remote Directories and Striped Directories

Technical Description and Approach

Vertical scale improvements from Project 1 aim to increase the performance of a single-node server. However, such efforts are subject to diminishing returns due to fundamental architectural constraints of a single server. A breakthrough in metadata scalability and throughput can be achieved by exploiting horizontal scale.

Distributed NamespacE (DNE) allows the Lustre namespace to be spread over many metadata servers. This ensures that both the size of the namespace and metadata throughput can be scaled with the number of servers. File and directory distribution can also be controlled to reserve and dedicate specific metadata resources for different sub-trees within the namespace.

This project is split into two subprojects in order to deliver some of the benefits before the full scope of DNE is implemented and to provide earlier feedback on real-world deployment and administration issues

Subproject 2.1: Remote Directories

This subproject distributes the Lustre namespace over multiple metadata targets (MDTs) under administrative control using a Lustre-specific mkdir command. Whereas normal users are only able to create child directories and files on the same MDT as the parent directory, administrators can use this command to create a directory on a different MDT. The contents of any directory remain limited to a single MDT. Rename and hardlink operations between files and directories on different MDTs return EXDEV, forcing applications and utilities to treat them as if they are on different file systems. This limits the complexity of the implementation of this subproject while delivering capacity and performance scaling benefits for the entire namespace in aggregate.

Metadata update operations that span multiple MDTs are sequenced and synchronized to create and/or increment the link count on an MDT object before it is referenced by the remote directory entry and to update the remote directory entry before decrementing the link count and/or destroying the MDT object it referenced. Although this may result in an orphan MDT object under some failure conditions, it ensures that the Lustre namespace remains intact under any and all failure scenarios. All other metadata operations avoid synchronous I/O and execute with full performance.

This subproject includes the implementation of OST FIDs (File Identifiers). These are required to overcome a limitation in the current 2.x Lustre protocol that would otherwise prevent a single file system from having more than 8 MDTs. Addressing this technical debt in the first subproject of DNE avoids protocol compatibility issues that would arise if this feature were implemented after Remote Directories were used in production.

Subproject 2.2: Striped Directories

Striped directories allow single directories to be distributed over multiple MDTs under administrative control to scale both the capacity and throughput of those directories. Distributed operations needed to operate on these directories are sequenced and synchronized as described above; however, file creation within such directories remains local to their directory stripe and, therefore, avoids synchronous I/O and executes with full performance. Distributed rename and hardlink operations will be supported so that they work as expected within a single striped directory.

Striped Directories remove the limit on the maximum number of entries in a single Lustre directory (currently 100 million for ldiskfs) and increases both the maximum single client open files limit and the peak rate at which multiple threads in a single client can create files in a single directory.

Technical Debt

The work that will be done for the Remote Directories and Striped Directories subprojects is targeted to the restructured OSD code base that will allow the DNE feature to operate on different back-end file systems. This allows a cleaner implementation and removes one layer of the old metadata stack. As mentioned above, it also implements OST FIDs which overcomes limitations of the Lustre protocol that would otherwise limit the maximum number of MDTs to 8.

Project 3. Lustre File System Checker

Technical Description and Approach

Contractor shall develop an online file system checker and scrubber (to be called LFSCK) that will maintain distributed coherency across the file system though consistency checks between MDT inodes, OST objects and hard links. For DNE file systems there are additional consistency checks for directory references across MDTs.

Subproject 3.1: Inode Iterator & OI Scrub

OI Scrub will implement a kernel process for traversal of all inodes on the MDT, and will verify that the FID attribute stored in each inode is correctly stored in the Object Index (OI). Any corrupt or incorrect entries are fixed to ensure that all the FID to inode mappings in the Object Index are correct. This is required for correct MDT operation after a file-level backup/restore.

The inode iteration will be controlled from userspace so that it can be launched periodically, or manually after the MDT is restored. The MDT device can also be configured to verify all OI lookups, and if it detects any inconsistencies during operation, it can automatically launch a full scan. It will also be possible to rate-limit the iteration rate to avoid impacting the performance of other file system operations.

OI Scrub provides the foundation for later subprojects of the distributed file system check.

Subproject 3.2: MDT-OST Consistency

MDT-OST consistency will implement functionality for distributed verification and repair of the MDT inode to OST object mapping. This will add additional functionality while the MDT is iterating over the inodes (see Subproject 3.1) to check the file layout (LOV EA) to verify that the objects referenced in the file layout exist and that each object has a back reference to the correct MDT inode. Incorrect or missing back pointers on the OST objects will be corrected, and missing objects will be recreated when detected.

The UID and GID of OST objects will also be verified to match that of the MDT inode to ensure correct quota allocation. After the MDT iteration is complete, any unreferenced OST objects will be linked into a lost+found directory.

Subprojects 3.1 and 3.2 together constitute a complete replacement of the existing LFSCK utility for local file systems. This will allow complete checking of non-DNE file systems while the file system is online.

Subproject 3.3: MDT-MDT Consistency

MDT-MDT Consistency will add concurrent distributed verification and repair for DNE file systems. This will add functionality while the MDT is iterating over its inodes to check that directory entries reference inodes correctly and consistently with each inode’s back-pointer to its parent directory. This includes cross-MDT references where the directory entry and inode are located on different MDTs. Incorrect back pointers and orphan inodes will be resolved when detected. This will allow complete checking of DNE file systems.

Subproject 3.4: Performance

This subproject will ensure LFSCK is ready to be used in production environments. It will characterize and optimize the performance of the features implemented in Subprojects 3.1-3.3, ensure that the performance impact of background scrubbing is sufficiently controlled, and determine whether Lustre protocol modifications (e.g. support for aggregate RPCs) are required. Administrative controls and monitoring will be finalized and documentation and procedures will be provided for system administrators. This subproject will also characterize e2fsck performance in order to understand, and possibly reduce, the time required in the offline e2fsck step needed to check ldiskfs OSDs.

Technical Debt

Subproject 3.1 restores the ability present for pre-2.x MDT file systems to be backed up and restored using normal file-level tools such as tar and rsync. Without LFSCK Subproject 3.1, Lustre 2.x MDTs can only be backed up and restored using device level tools, such as "dd", that copy the entire device.

Completion of Subproject 3.2 eliminates the need to use the old stand-alone lfsck tool to check the distributed Lustre state. The old lfsck tool is slow to use, unusable at large scales and causes a significant burden on the maintenance of the Lustre-patched e2fsprogs due to the need to link in an external database package. The old lfsck is not portable to different back-end file systems whereas the new lfsck tool will be independent of the back-end file system type.

Project Approach

Technical Approach

Contractor shall use a rigorous development process, which focuses on the early processes in the software development lifecycle (SDLC) to properly define the requirements and solutions early in the process. By focusing early in the project on the Scope Clarification, Solution Architecture, and High-Level Designs (HLD), Contractor shall define proper solutions early, eliminate defects early, and keep OpenSFS better informed throughout the process.

Management Approach

This section identifies the standard processes used when developing any of the subprojects, and will be used for each of the subprojects listed above. If there is a planned variation to these steps, for example when a Solution Architecture document is not necessary, this is explicitly identified below.

OpenSFS Lustre Development

Project Milestone Detailed Overview

1. **Scope Statement:** Contractor shall provide a brief (2-3 pages) summary of their understanding of the problem statement and resulting project scope to be reviewed by the OpenSFS Project Approval Committee (PAC) and submitted for approval or rejection by the Contract Administrator (CA). This statement shall include:
    a. Problem statement
    b. Statement of high-level project requirements/goals to be satisfied
    c. Statement of in-scope and out-of-scope work for project
    d. Statement of project assumptions or constraints
    e. Statement of key deliverables and milestones

2. **Solution Architecture:** Contractor shall provide a document that outlines requirements, use cases, and a solution framework in an effort to address items defined in the scope statement. The Solution Architecture shall be reviewed by the PAC and submitted for approval or rejection by the CA. This document shall include:
    a. Identification of requirements that address the subproject requirement
    b. Defined list of use cases for Test Plan
    c. Defined list of Acceptance Criteria-the acceptance criteria will be defined for the entire subproject (excluding the Delivery task) and are measured during the Demonstration Milestone.
    d. Proposed high-level solution that addresses the subproject requirements

3. **High-Level Design (HLD):** Contractor shall document a recommended solution that addresses the subproject requirements. This document shall describe how the solution will work including basic protocol structures as applicable. The proposed solution shall be documented with the elements below and reviewed by the PAC and submitted for approval or rejection by the CA:
    a. Description of the solution elements/implementation components and how they will work
    b. Explanation of why/how the proposed solution will address the subproject requirements
    c. Identify any risks or unknowns with the proposed solution
    d. Define API and protocol changes, if applicable
    e. Estimation of engineering and unit testing effort needed to complete the solution
    f. Prototype code for the solution, if complexity requires

4. **Implementation**: Contractor shall complete implementation and unit testing for the approved solution. Contractor shall regularly report feature development progress including progress metrics at project meetings and engineers shall share interim unit testing results as they are available. OpenSFS at its discretion may request a code review. Completion of the implementation phase shall occur when the agreed to solution has been completed up to and

including unit testing and this functionality can be demonstrated on a test cluster. Code Reviews shall include:

a. Discussion led by Contractor engineer providing an overview of Lustre source code changes
b. Review of any new unit test cases that were developed to test changes

5. **Demonstration** Upon functional completion of the feature, Contractor shall demonstrate the appropriate functionality of the subproject. This shall be done through execution of test cases designed to prove the acceptance criteria defined during the Solution Architecture. Demonstration specifics will be defined and mutually agreed to for each subproject in the scope and architecture phases.
    a. Functional Test Plan: Contractor shall develop and recommend a functional test plan, as defined by OpenSFS, designed to demonstrate the functional completeness of the feature. The results of functional testing with supporting documentation will be presented to OpenSFS for review.
    b. Performance Test Execution: Contractor shall define and recommend a set of performance tests as defined by OpenSFS to document the performance characteristics for performance related features. Contractor shall execute these tests and present results of these tests to OpenSFS for review.. OpenSFS shall provide adequate test platforms when scale is necessary for performance testing as recommended by Contractor and defined by OpenSFS.

6. **Delivery**: Contractor is planning for annual feature releases and quarterly bug-fix releases. Contractor shall maintain its own Lustre repository that will be open to the community. All features and fixes from Contractor will be submitted to the community for inclusion in the canonical Lustre tree. All software development within the scope of this agreement shall be conducted in Lustre repository that is open to the community. Contractor’s completion of this milestone shall be based on Contractor’s integration of the project development branch into the Contractor Lustre tree and integration of these features into a standard Contractor supported release.

Acceptance by OpenSFS of Contractor’s completion of each subproject shall be based on Contractor’s integration of the project development branch into the Contractor Lustre tree and integration of these features into a standard Contractor supported release. This release will then be tested within four (4) weeks or 20 business days, by OpenSFS (performance, stability, feature set compliance) for final acceptance or rejection. No written response from OpenSFS shall constitute implicit acceptance and approval. Contractor will land the project branch to the tree following internal landing practices and policies.

Project Approval Committee

OpenSFS shall designate a Project Approval Committee (PAC), comprised of community members that shall have final and documented authority for project-level decision making. The purpose of the PAC is to both facilitate clear decision-making throughout the project and provide an avenue for community collaboration during the project. Timely decision-making will support the advancement of projects and ensure that OpenSFS is well informed about project milestone progress. The OpenSFS board-appointed Technical Representative shall serve as committee chair of the PAC and shall have final authority on all decisions of the PAC subject to the discretion of the OpenSFS board.

OpenSFS shall openly declare the membership of this committee and shall only change this membership during the course of the project with advance notification to Contractor.

Milestone-Task Approvals

A milestone-task is defined as the work or effort necessary to complete a specific project milestone, as described in *Project Milestone Detailed Overview* sub-section within this document, and is a sub-component or step along the way to completing a sub-project or project within the overall contract.

Approval will be required from the OpenSFS Contract Administrator in writing (email is acceptable) before Whamcloud starts work on a milestone-task.

Approval will be required from the Contract Administrator at the end of each milestone-task, for Whamcloud to invoice for the milestone.

For all requested approvals, the Contract Administrator, will have 2 weeks (10 business days) from work submission or decision request to provide feedback and render a decision. No response from OpenSFS shall constitute implicit acceptance and approval. If at a later time there is a dispute by OpenSFS, a separate resolution meeting will be held between OpenSFS PAC and Whamcloud Senior Leadership to include the CEO and CTO.

Subproject Termination

The Contract Administrator reserves the right to terminate any subproject at any time. In the event of termination of a subproject the Contractor shall terminate work and payment shall be made, if applicable, in accordance with the Terms and Conditions. Milestone will be considered complete once completion is recommended in writing (email is acceptable) by the PAC and approved by the Contract Administrator.

OpenSFS Lustre Development

Project Planning

Projects contained within this SOW are comprised of Research and Development engineering effort and as such the solutions are not necessarily straightforward or easy to determine. Contractor shall make best efforts to define appropriate target delivery dates based on information available.

With the completion of each milestone, Contractor shall deliver updated estimates for both interim milestone delivery and effort estimates using data and information acquired during the process of the project to refine estimates.

If, during the course of a Research and Development project, information is discovered that was previously unknown and significantly complicates the solution, Contractor shall present this new evidence to OpenSFS with a list of proposed alternate solutions. Some solutions may become so large as to not be able to be completed under the fixed price of the contract. In this event, Contractor shall offer alternate options from which OpenSFS may select that can be completed within the bounds of the contract.

Reporting

Contractor shall submit a monthly progress report in a format as provided by Contractor and reasonably acceptable by OpenSFS no less than two business days after the last day of each month highlighting progress made on each deliverable. Contractor shall provide documentation of completion as defined in Project Milestones Overview for each deliverable.

Project Meetings

Weekly project calls shall be held with participation from both Contractor and OpenSFS project team members. The purpose of this meeting is for project team members to gather to discuss progress against plan. OpenSFS attendance shall include at least one PAC member. Contractor attendance to include Senior Engineer assigned to project, Project Manager assigned to project, and any relevant development engineers depending on topic(s) of discussion. Contractor shall provide a standard project review template as recommended by Contractor and defined by OpenSFS to include progress since the last meeting, current activities, risks and issues. Meeting minutes shall be recorded at all project meetings and stored as project artifacts.

Software Licensing and Source Code

All software developed under this contract shall be submitted under the terms of the OpenSFS contribution agreement.

OpenSFS Lustre Development

Lustre Community Development

Contractor shall submit all GPL Lustre software changes (including test cases) to the community canonical Lustre code base. It is at community's discretion to incorporate Contractor's changes into the canonical Lustre tree. Contractor shall maintain a separate code repository and make this available to the broader Lustre community.

Lustre Software Enhancements

All changes to Lustre software shall be made against a production release of Lustre 2.x for this proposal. Changes made to Lustre as part of this contract will break interoperation between Lustre 2.x servers and Lustre 1.8.x clients. No development or changes will be made against the Lustre 1.8.x or 1.6.x code base. Contractor shall independently test and compile all modifications against a production release of Lustre (Production Lustre Release + Contractor patches).

Vendor Neutrality

All software shall be designed and developed to be broadly applicable to the entire Lustre. Contractor shall not design or implement to benefit or exploit a particular hardware configuration. The Lustre software has always been designed to work on a wide variety of vendor server and storage platforms; Contractor shall not change this fundamental Lustre design principle.

OpenSFS Furnished Access and Hardware

OpenSFS or its constituent members shall provide access to a test cluster (configuration to be recommended by Contractor and defined by OpenSFS) that will be available throughout the duration of Projects 1 & 2. Contractor shall be notified of any changes to this test cluster throughout the duration of Projects 1 & 2.